ESD-TDR-64-159

SR-126

THE COLINGO SYSTEM DESIGN PHILOSOPHY

TECHNICAL DOCUMENTARY REPORT NO. ESD-TDR-64-159

NOVEMBER 1964

J. F. Spitzer
J. G. Robertson
D. H. Neuse

Prepared for

473L/492L SYSTEM PROGRAM OFFICE

ELECTRONIC SYSTEMS DIVISION

AIR FORCE SYSTEMS COMMAND

UNITED STATES AIR FORCE

L. G. Hanscom Field, Bedford, Massachusetts

Project 492

Prepared by

THE MITRE CORPORATION
Bedford, Massachusetts
AF Contract 19(628)-2390

THE COLINGO SYSTEM DESIGN PHILOSOPHY

TECHNICAL DOCUMENTARY REPORT NO. ESD-TDR-64-159

NOVEMBER 1964

J. F. Spitzer
J. G. Robertson
D. H. Neuse

Prepared for

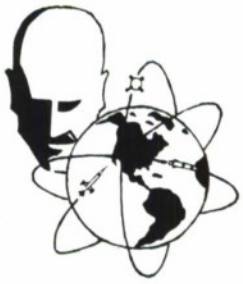
473L/492L SYSTEM PROGRAM OFFICE

ELECTRONIC SYSTEMS DIVISION

AIR FORCE SYSTEMS COMMAND

UNITED STATES AIR FORCE

L. G. Hanscom Field, Bedford, Massachusetts



Project 492

Prepared by

THE MITRE CORPORATION
Bedford, Massachusetts
AF Contract 19(628)-2390

THE COLINGO SYSTEM DESIGN PHILOSOPHY

ABSTRACT

This report describes the design and operation of COLINGO (Compile On-LINe and GO), a program system embodying a computer control and query language that provides the operator with a grammar and vocabulary approximating English to control program data and equipment in a data processing system. COLINGO provides both logical and mathematical control of data in the system, and allows the operator to specify the output format and content. Additionally, it provides the capability of on-line programming in a higher-ordered language called the COLINGO Control Language (CCL). The system provides internal editing, detection, and error correction features.

A primary design objective was to make COLINGO adaptable to a changing set of requirements and particularly responsive to data compatibility, on-line programming, simple extension, and good man-machine interface. This necessitated maintaining independence of data from programs so that changes in one do not affect the other. The objective was achieved by means of a common retrieval and control program that uses a set of dictionary tables as the translation medium between programs and data. With this concept, changes in programs or data are reflected in modifications to the tables instead of the usual modification to the data or programs, or both.

REVIEW AND APPROVAL

Publication of this technical documentary report does not constitute Air Force approval of the report's findings or conclusions. It is published only for the exchange and stimulation of ideas.

EMMETT V. CONKLING
Colonel, USAF
System Program Director
473L/492L System Program Office
Deputy for Command Systems

CONTENTS

LIST OF ILLUSTRATIONS

INTRODUCTION

Our objectives in designing COLINGO were to plan and design from the start a system that could grow and be easily modified based on the user's changing operational requirements, and the designer's technical experience. We knew that neither group at the onset could state the exact requirements, design characteristics, or desired capabilities that would bear on the system's configuration at future points in time.

We envisioned a developmental effort in parallel with the operational activity so that longer range design and development could be conducted without jeopardizing operational activities, while short-range changes and improvements could be easily made to the operational system as the need arose.

Finally, we felt that the user of the system should have products available at all times and confidence that his immediate problems had a good chance of solution by a means of on-line programming techniques or would certainly be included in the longer range development effort.

These objectives and associated constraints dictated the following general design goals for COLINGO -- COLINGO was to be:

- A system that would accept most types of data from most sources, with minimal reformatting.
- A system that would perform most any logical or mathematical manipulation of the data.

- A system that would allow rapid updating without imposing difficult constraints on the update format.
- A system that would allow for on-line programming to meet many day-to-day problems, and one that would allow low lead-time off-line programming for more complex problems.
- A system with simple program maintenance and extension capabilities, to better allow the using command to participate and guide the evolutionary design of the system.
- A system with a good man/machine interface and control language, which would be data compatible with most associated organizations.

In keeping with these design goals, MITRE undertook development of a Basic Program Set to store, retrieve, and manipulate data and to serve as a framework and nucleus for expansion. The system was to maintain independence of data from programs so that changes in one would not impact the other. This was to be achieved through a common retrieval and control program that used a set of dictionary tables as the translation medium between programs and data. With this concept, changes in programs or data would be reflected in modifications to the tables instead of the usual modifications to the data or programs, or both.

The controlling element in the design was to be a small in-core executive program which would interpret a query and control language - the COLINGO CONTROL LANGUAGE (CCL).

THE COLINGO CONTROL LANGUAGE

CCL was to exist in two forms - a basic query language subset which could be learned in a few days, and a limited programming language which would provide capability approaching that of the common compiler lanugages.

CCL was to provide an operator interface independent of the computer equipment configuration (initially an IBM 1401), and was to be as insensitive as possible to equipment additions, except for a betterment in response time. This suggested a translation or interpretive language, where words, punctuation, and grammatical structure would activate programs which would come into use in the sequence they existed in the input message.

The COLINGO Message

The man/machine communication vehicle in the COLINGO system was chosen to be a near-English language. All communication with the computer was to be accomplished by a CCL message (sometimes called a query, query statement, or simply statement). A message is of the type illustrated by Fig. 1. The intent of the CCL message is to provide control over the entire system (i.e., data, programs, equipment, and output) through a user oriented language. COLINGO is a bare system of computer programs called by the CCL interpreter; it involves no data, and includes no inherent operational capabilities. It is applicable to almost any data management task, but is entirely dependent upon the data base specification and operational requirements that the particular problem dictates.

The CCL has a grammar, punctuation, and vocabulary which must be strictly adhered to in phrasing a message. The vocabulary is, however,

GET A-FILE IF STRENGTH/AUTH GR 500

EXECUTE 01 02 03 IF/NOT PRINT ALL.

(Underlined words are action verbs - all words following action verb n up to the next action verb n + 1 are parameters of action verb n).

(The parameters following the EXECUTE action verb are the labelled locations of other messages (stored on disk QUIC files)).

(Underlining of action verbs is for clarity only; they are not actually underlined in the message entry).

Fig. 1. Typical Message Format.

to a great extent created by the user to fit his particular needs. In this respect, COLINGO is not unlike the compiler languages, where data categories can be assigned names of the user's choosing. This ability, along with an English grammar and punctuation, provides a language capability applicable to a wide variety of applications.

In practically all data management tasks, capability is required in the following areas:

(1) The data qualification area (logical processing)

(2) The data computation area (mathematical processing)

(3) The output area

In keeping with the first requirement, COLINGO provides a logical processing verb (IF), which allows rather complete Boolean algebra over the data records. As an example of this capability, take a COLINGO message in a typical military environment:

GET AIRFIELDS-FILE IF COUNTRY EQ US AND RNWY/LENGTH GR 8000.

With respect to the second requirement, COLINGO provides a mathematical processing verb (COMPUTE), which accommodates a number of the common mathematical processes. For example:

GET FORCE-STATUS-FILE IF UNIT EQ 82-ABN COMPUTE FORCE-RATIO =
TOTAL-OFFICERS / TOTAL-EM.

Finally, in the output area, COLINGO embraces several verbs (PRINT, PUNCH, TYPEOUT, REPORT), which provide capability to a number of peripheral devices in a number of different formats. For example, the results of the AIRFIELDS-FILE message could be output to the printer on two data categories by the message:

PRINT NAME LOCATION.

In addition to the above capabilities which are recognized as necessary in all data management problems, another capability is extremely important in military data management, namely, an update capability. This is accomplished by two verbs (UPDATE and CHANGE). For example:

GET MISSILE-FILE IF TYPE EQ AJAX OR ZEUS CHANGE RANGE TO UPDATED-RANGE.

The COLINGO designers recognized that a query language and programming language need not be separate entities. Indeed, there were many applications where the uniquely rapid response of a query language would be valuable in a programming application.

Two design decisions allowed the accomplishment of this combined language capability. They were:

(1) A sequence control verb (EXECUTE).

(2) A set of out-of-core control files (QUIC* files), providing extension of the normal in-core message area (limited to 467 characters). These control files provided a multiple chained message capability (called "message strings"), with the EXECUTE verb providing the message links and their sequence (or "schedule").

The EXECUTE verb provides a reference to the Executive routine, setting in this routine the labels of a series of messages stored on disk or tape. Additionally, any message in the string can change or reset the Executive schedule.

---

*Self-describing files of 7000 characters each, normally stored on disk, and characterized by organization conducive to rapid retrieval of the information stored therein.

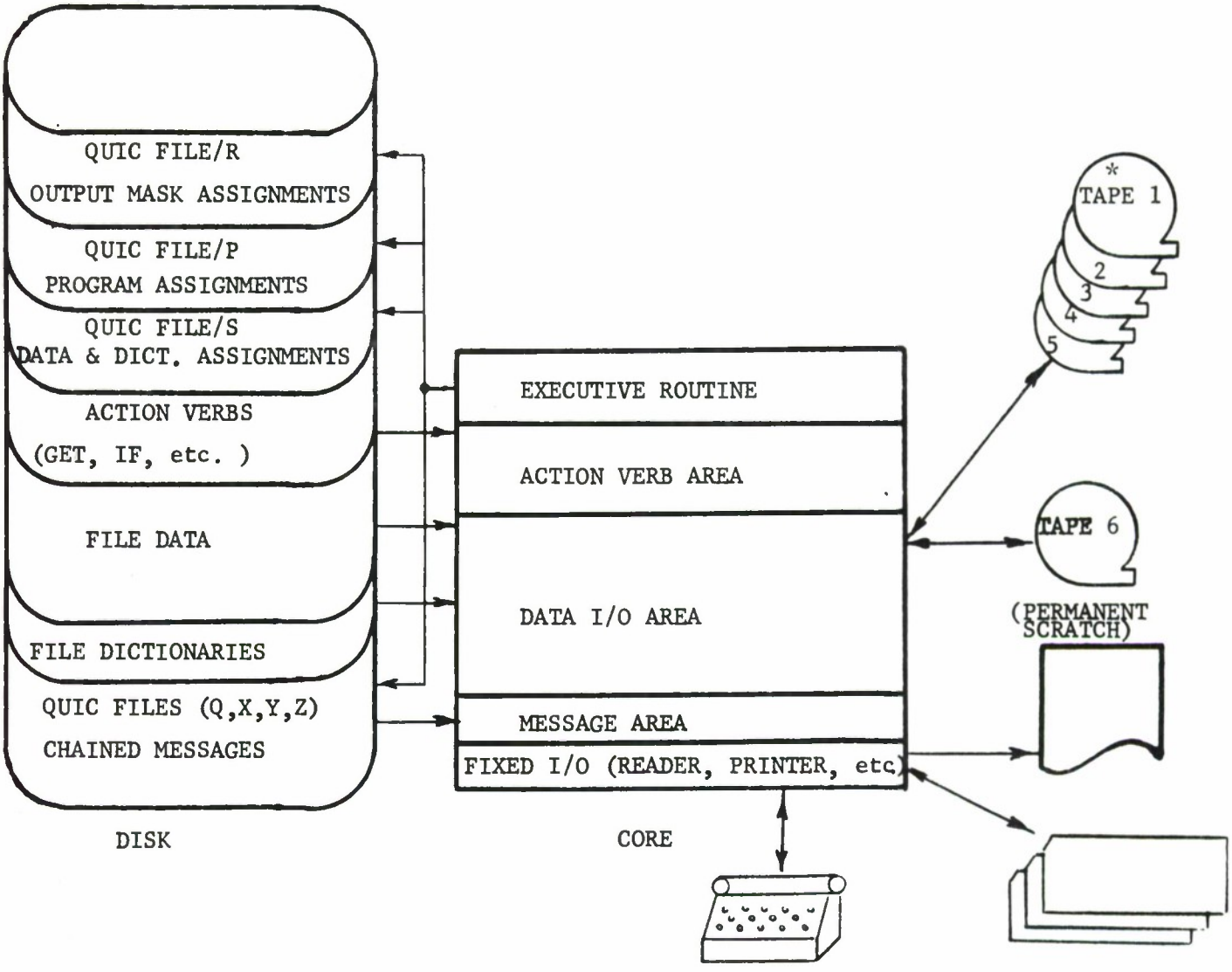
A message string is in theory unlimited in length, and is furthermore modifiable based on data decisions, operator interventions or message composition or modification programs.

The message string is the underlying power of the COLINGO system, providing an extremely practical message concept on a small machine. It enables the sequential execution of a string of any number of programs, with only one program in core at a time.

COLINGO Program Operation

It was earlier mentioned that COLINGO is a system consisting of programs only, with no data imbedded therein. How, then, does the data interact with the programs? This can be best illustrated by Fig. 2, where disk, tape, and core are represented as repositories for programs, data, and program-data interaction, respectively. Notice that the 1401 core has room for only one action program and only one data record (and its describing dictionary) to exist simultaneously. The action program area is labelled the "action station", the data record area the "I/O station". Still a third area of 467 characters is the "message station", a fourth area is the Executive routine. The Executive routine is the only permanent in-core routine, and comprises the upper 500 characters of core.

The disk is the usual program storage media (although tape can be used with degraded response) from which a program can be called to the action station in core. The calling of the program is accomplished by the Executive routine scanning the message area for an action verb and calling the associated program module. The parameters following the referenced action verb describe to the program module the instructions for performing specific actions on the incoming data records.



*TAPE 1 may be used in place of the disk to contain programs & QUIC Files.

Fig. 2. COLINGO Program Structure.

The Executive routine contains a "schedule" area into which an EXECUTE verb loads the labels of a series of messages stored on disk. The Executive routine then calls the messages at these labels in sequence. For each message in the message area, the Executive performs a left to right scan for action verbs. When an action verb is encountered, the associated program is called into the action station area in core, where it awaits the data records. The parameters supplied to it describe the process variations that it can perform on the data. Based on the data and/or parameters, the action program can advise the Executive routine of a change in schedule or a change in parameters for the next message on the schedule.

Because of the limited core, only one action station can exist at any instant of time. Hence, only one action program is active at any one time. This requires intermediate out-of-core storage facilities for data records between action verbs, and this storage is afforded by a scratch tape. An action program exists only long enough for the program to complete its function, and then that program is destroyed by the next program being read in.

A small core mapped in Fig. 3 forced this single station approach on COLINGO "D"; obviously in a larger core, many programs could be available simultaneously to work on the data. Even assuming a larger core (such as the 1410 could provide), it will in general, not always be possible to assemble all programs in core simultaneously. The sequenced program operation will be retained in COLINGO 10 (the proposed COLINGO for the IBM 1410) for:

(1) those messages whose total action verb storage exceeds core, and

CORE ALLOCATION	FUNCTION
1-1199	FIXED I/O Area + Indicators
1200-1999	Stacks (2) for Parameters/Pgm
2000-7999	Work Area (Dictionary + Data)
8000-8999	Data Stack
9000-10999	Sub-Executive Routine
11000-14449	Action Verb Program Area
14500-15449	Executive I/O Area
15500-15999	Executive Routine

Fig. 3. COLINGO Core Map.

(2) those action verbs in a message which are assigned to the same location (station) in core.

In this latter application, it should be mentioned that the COLINGO action verbs are assembled between absolute core limits (i.e., they are not "relocatable"). For the COLINGO 10 system, several action stations will be available, and certain action verbs will be assembled relative to these several action stations. Relocation will not be permitted, and if two action verbs are assigned to the same location, the sequential method will still be required.

COLINGO Data Structure

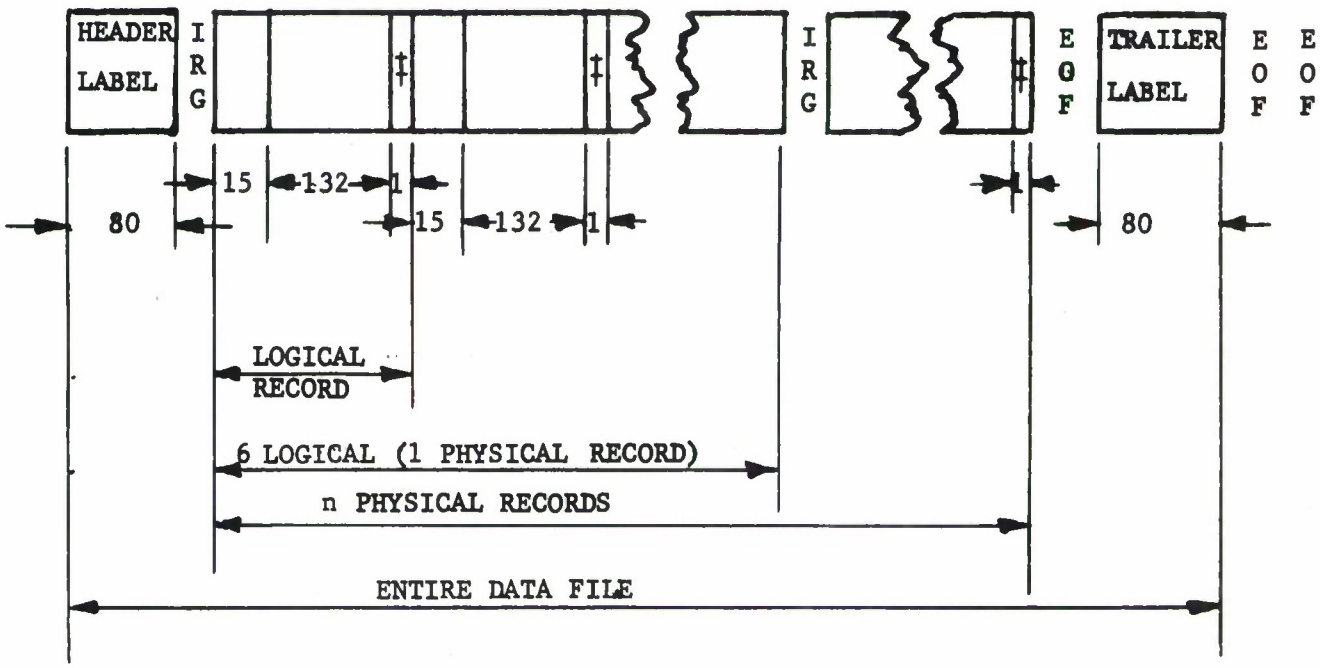
In order for COLINGO to process data, the data must be generated and described. A generation program is provided to generate many files contained on cards, card images on tape, or fixed-length unit records on tape to COLINGO format. Restrictions are as follows: The tape unit record length must not exceed 983 characters and must not be less than 25 characters; file generation automatically appends 16 control characters to each logical record. In the normal case, the card record files cannot exceed 12 different format types (i.e., there can be no more than 12 different types of cards per data file). If all information categories about a data entry (object) are single-valued, one logical record is sufficient to carry all information about the object. This logical record is called a Master record. If, however, certain information categories associated with an object contain multiple values, one or more additional logical records are associated with the object to carry the multiple values. These are called Trailer records. The information category carrying the most multiple values determines the number of

logical trailer records. If this particular category carries n values, there are n-1 trailer records and one master record. All logical records (master and associated trailers) are the same length, and are automatically blocked to form physical records as close to or equal to 999 characters as possible. Fig. 4 is an example of this blocking process as applied to an input file of 132 character tape unit records. Note that after the 16 control characters are added to each logical 132 character record, the resulting physical record length is 888 characters (148 x 6), consisting of 6 logical records. The drawing illustrates that the first record in the file is an 80 character header record, and the file is terminated by a separate file of 80 characters comprising the trailer record. This type of file meets the IBM IOCS[1] Form 2 requirements. The 132 character record size is for example, only; any logical record size between 25 and 983 characters is legal input.

COLINGO Dictionary Structure

After a file has been generated, it must be described. The division of the logical record into fixed length fields, and the property-names given these fields, is the function of the describing dictionary. While the composition of the data itself indicates the normal field sizes and descriptions, it is possible to describe the logical record with any dictionary which properly describes the length of that record. Furthermore, several dictionaries (maximum of 36) may describe one file, and in addition the dictionary description may be altered on-line by the operator, further extending the multiple definition ability.

A dictionary is generated by creating a COBOL[2] Data Division similar to Fig. 5, which describes the file being processed. This Data Division is a set of cards formatted in accordance with COBOL requirements. Presently, only the "PICTURE IS" clause has been implemented for data



IRG = inter-record gap

EOF = end-of-file

‡ = record-mark

Fig. 4. COLINGO File Structure.

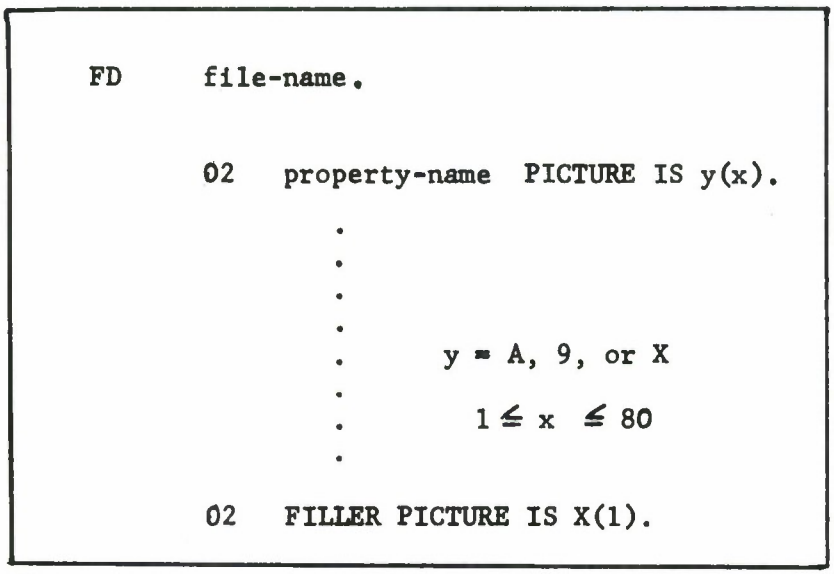


Fig. 5. COBOL Data Division.

description in COLINGO. In the Data Division are described the logical breakdowns of the data (into levels, e.g., 02, 03), the nature of the data (numeric, alphabetic, or alpha-numeric), and the reference or data-names (property-names) assigned to the data fields. From this information in the COBOL Data Division, the COLINGO dictionary generation program creates a describing dictionary record and assigns this particular dictionary record to the data file of the same name. This procedure can produce several dictionaries to accompany a single file -- perhaps several for query use, several for update use, and several more for output use. The various different dictionaries may be called by appending to them a single suffix number or character. In addition, certain action verbs (e.g., IF) can be modified by terms attached to their property-name parameters. These modification terms further modify the dictionary description of the appended parameters. As an example of this process, the logic level, data classification, field length, and property-name are illustrated in Fig. 6 as they appear internally to the COLINGO logic interpreter. To accomplish an on-line change to the normal describing dictionary, a parenthetical term may be appended to the property-name to change the logic level, field limits, and data class, e.g., IF NAME (03,9,+5,-1). This would qualify the NAME field (normally an 02 level, alpha-numeric, 15 character field) as a numeric, 03 level field, looking at only the 5th character. This is a <u>transient</u> change, in that the describing dictionary is not altered--only the internal compare logic is altered. A property-name which is not modified by a parenthetical term uses the normal describing dictionary items for level, class, and field definitions.

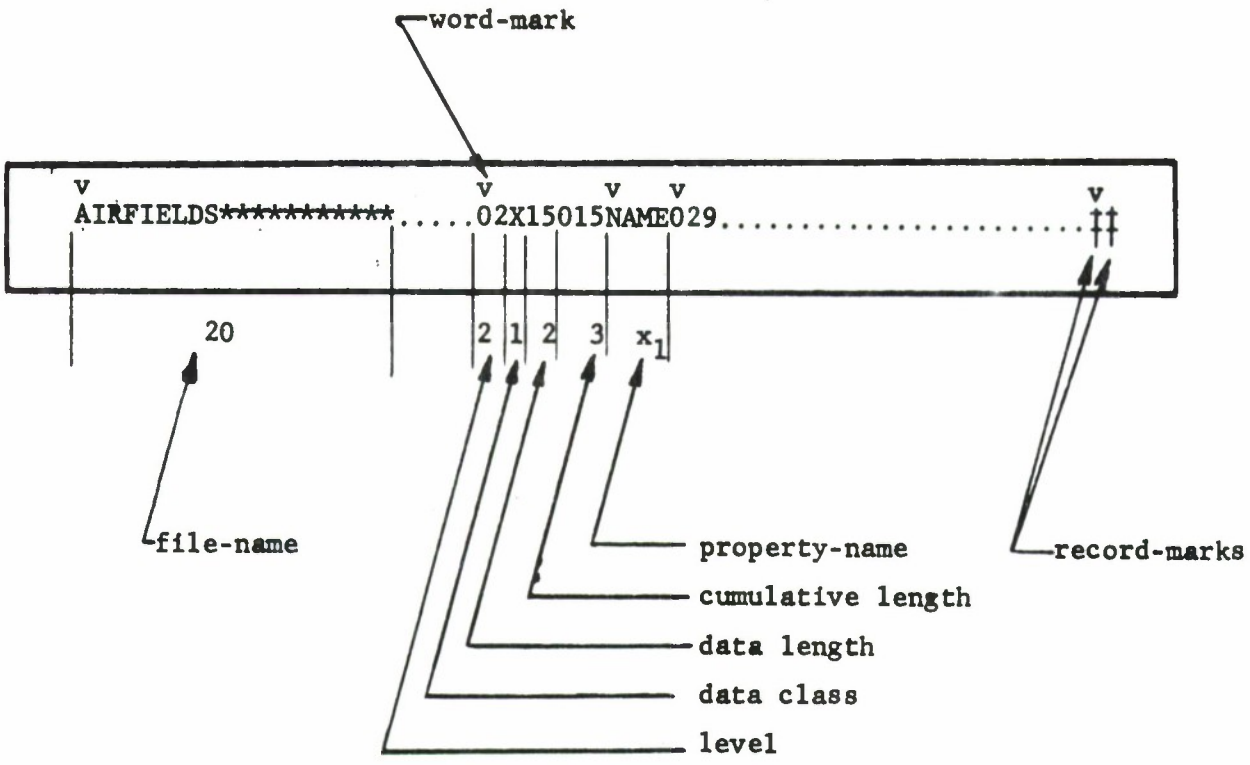


Fig. 6. COLINGO Dictionary Format.

It is the area of dictionary redefinition during execution that gives the COLINGO interpretive approach a singular advantage over a conventional compiler. Instead of planning all possible data breakdowns in advance (as one must in a compiler), COLINGO permits redefinition of the data area during exectuion, based on operator, data or program decision points. Each dictionary is essentially a mask overlay which can subdivide a file data record as prescribed by the overlay. Thus every character combination in a data record is accessible on-line by the combination process of storing certain dictionaries and/or using the modification terms available.

With the data generated and described by a dictionary, we are ready to process the data file.

COLINGO Data Operations

An all-master file (a file with only master records) is particularly easy to process, as there are no multiple values in any record, and the data structure has essentially one dimension (length). In this case, the retrieval logic needs to work on only one logical record per object (the master record), and in general any logical combination of property-names within this record can be used to qualify such a record. In this application, the COLINGO logic is compatible with COBOL qualification logic.

As mentioned before, in the case of a file having objects with multiple values on certain property-names, trailer records are required. This requires the COLINGO logic to work on several logical records per object (master + associated trailers) to qualify an object. Since the "OCCURS" clause of COBOL has not been implemented in COLINGO, the normal

COBOL 02, 03 level concept has been modified to handle the row-column matrix occasioned by a two dimensional data structure. In general, 02 defined property-names are qualified in the vertical dimension (column qualification), while 03 defined property-names are qualified in the horizontal dimension (row qualification). Logic on joint 02, 03 property-names occassions a combination of those respective logics.

To understand the operation of this two dimensional logic, let us take as an example the common chessboard shown in Fig. 7. To define the eight value categories we shall use the standard chess notation of property-names from left to right; QR, QN, QB, Q, K, KB, KN, KR. In this notation, the left-most column is the Queen's Rook column, the next the Queen's knight column, etc. The 1st row comprises the master record, and the value categories will be referred to as QR1, QN1, etc. The 2nd row value categories will be similarly referred to from left to right as QR2, QN2, etc., and will comprise the first trailer record. The 3rd through 8th row value categories will be labeled accordingly, and will comprise the 2nd through 7th trailer records. Any value category can assume any of the following legal values: KI, QU, R, N, B, P (the King and Queen have not been given their usual notation, K and Q, as this would have defied the rule that prohibits a value from being identical in spelling to a property-name). Since the COLINGO language provides no implicit way to identify rows, we will add a 9th column with the property-name ROW-ID. The values inserted in this column will be 1 in the master record and 2 through 8 in the seven trailer records.

The chessboard serves as an example illustrating the use of the 02, 03 logic in matrix retrieval. It shows how a two dimensional data matrix may be described. For example, let us test a row-column intersection

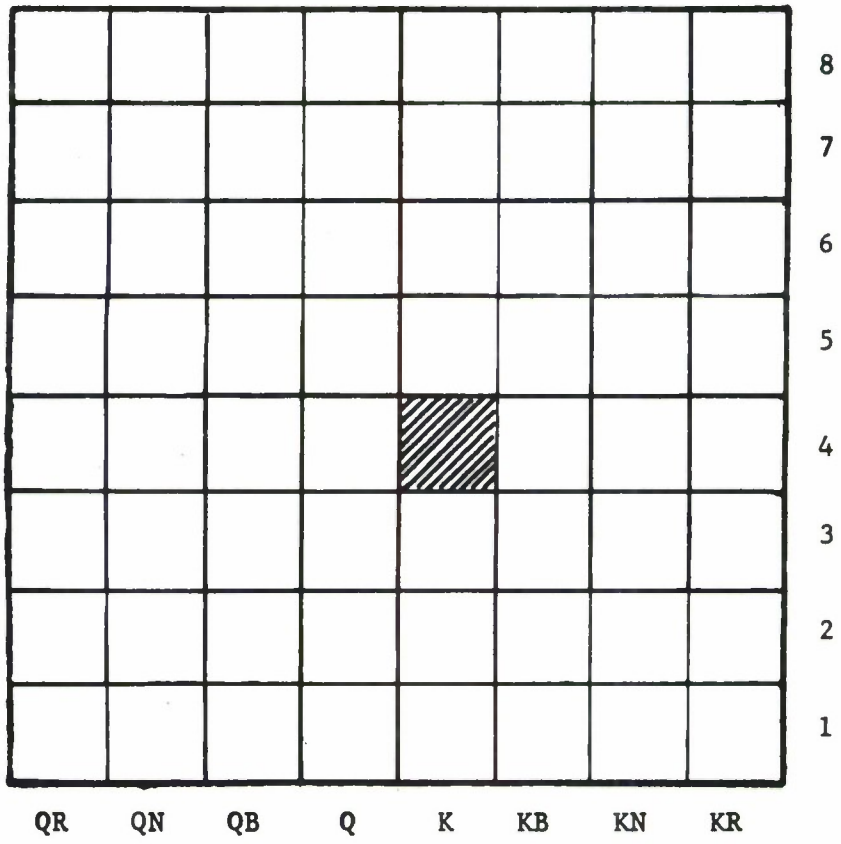


Fig. 7. Matrix Qualification.

(K4) for vacancy.

Query:

IF ROW-ID (03) EQ 4 AND K (03) EQ BLANKS.

Notice here the use of the modification term (03), to provide a transient modification of the property-names to 03.

This two dimensional data structure is referred to as a Data Record Matrix. Obviously, if the vertical dimension is 1, we have the degenerate case of all Master (or one dimensional) file. Since there is a single COLINGO data structure, there is a single logic program. Additionally, all COLINGO programs are designed to operate with a single data structure, thus permitting a high level of user isolation from the structure, if he desires such isolation. If isolation is not desired, as in on-line file generation and recombinations, the data record matrix may be compressed, expanded, or deleted in either dimension, at the control of the operator. In practice, a one dimensional data structure (all Master records), permits almost complete operator isolation from the data structure; this isolation is never possible in the two dimensional case (Master and Trailer records).

Let us now attempt some further interrogation of the chessboard.

Does row 4 of Fig. 8 have all Pawns?

The Query statement is:

IF ROW-ID (03) EQ 4 AND P EQ KR (03)

AND KN (03) AND KB (03) AND K (03) AND Q (03)

AND QB (03) AND QN (03) AND QR (03).

This is an example of row (03) logic.

Does the King's Rook column in Fig. 9 have at least one pawn?

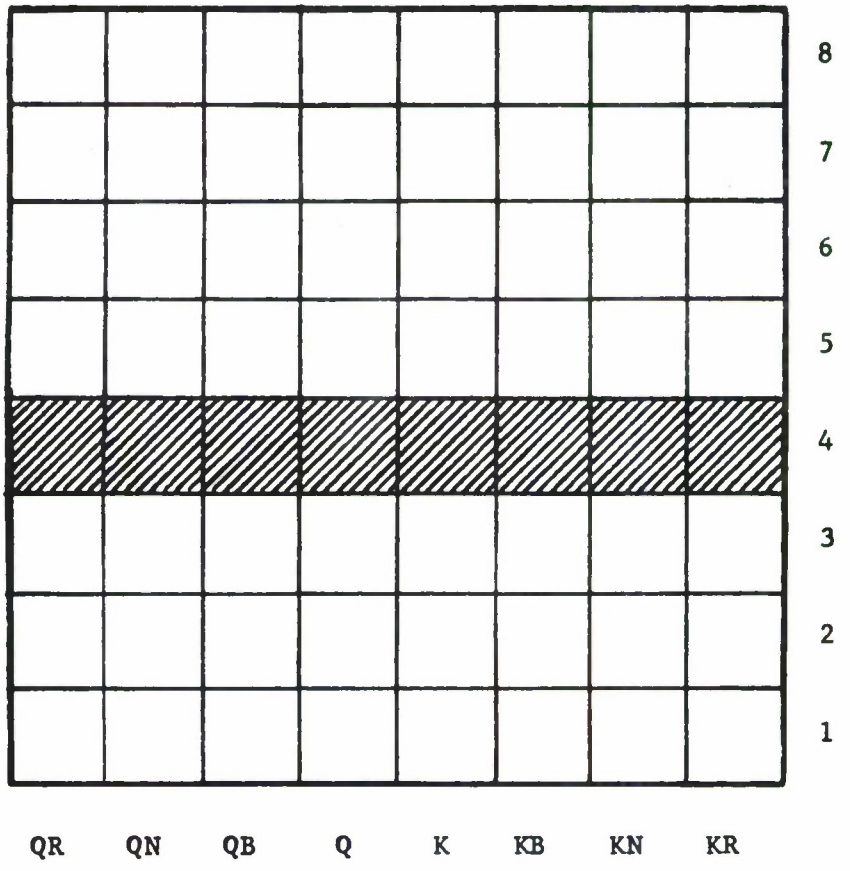


Fig. 8. Row Logic

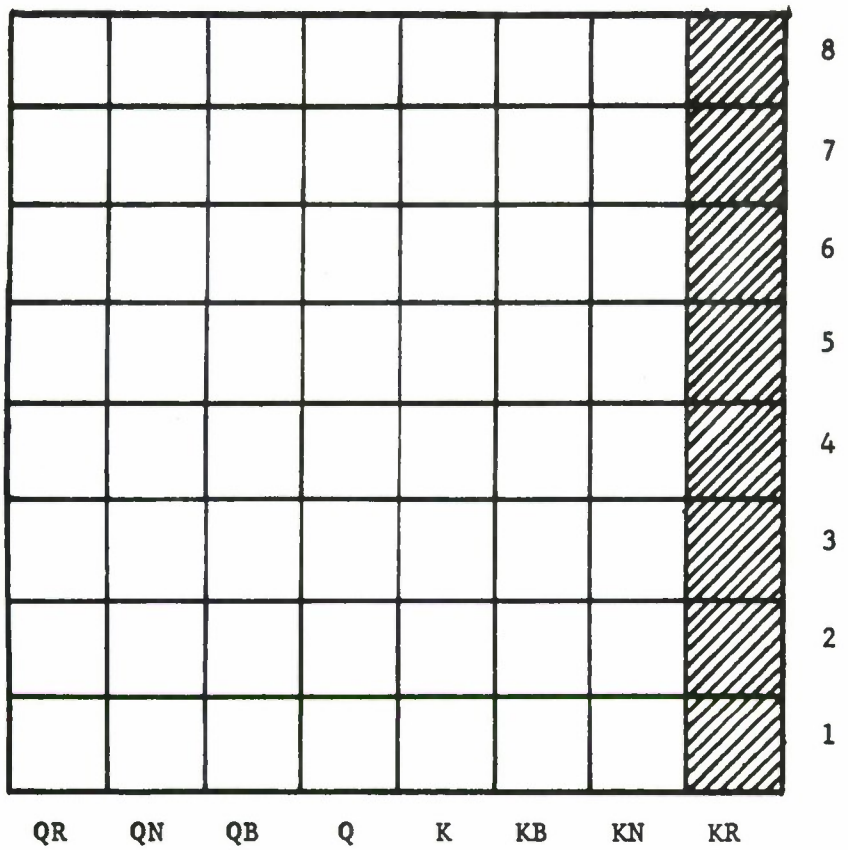


Fig. 9. Column Logic.

The Query statement is:

IF P EQ KR (02).

This is an example of column (02) logic.

In Fig. 10 is either King still in the King column (row unimportant), and are the white Bishops still in their original positions?

The Query statement is:

IF K (02) EQ KI AND KB (03) EQ B

AND ROW-ID EQ 1 AND QB (03) EQ B.

This is an example of combination column-row (02, 03) logic.

COLINGO Programming Procedures

Although the operator doing routine querying of the data base need not know the internal data flow, such knowledge is required for the person using COLINGO as a programming language. As a means of understanding this, consider the fact that COLINGO action verbs in the input message are processed sequentially, with intermediate and final results being output to tape. In all cases, result output is in the self-describing form (i.e., the dictionary that describes the data accompanies it). It is then necessary for any program (action verb) that requires access to this data to have a dictionary decoding module. This module essentially relates the parameters following the action verb to the data, using the dictionary as the data description medium.

It is, of course, possible (within limits) to allow an action verb to output a special formatted file that is not self-describing, with the presumption that the output format, data content, and data organization can be processed by the next action verb without description (i.e., that action verb requires a standard format input, and the prior module is

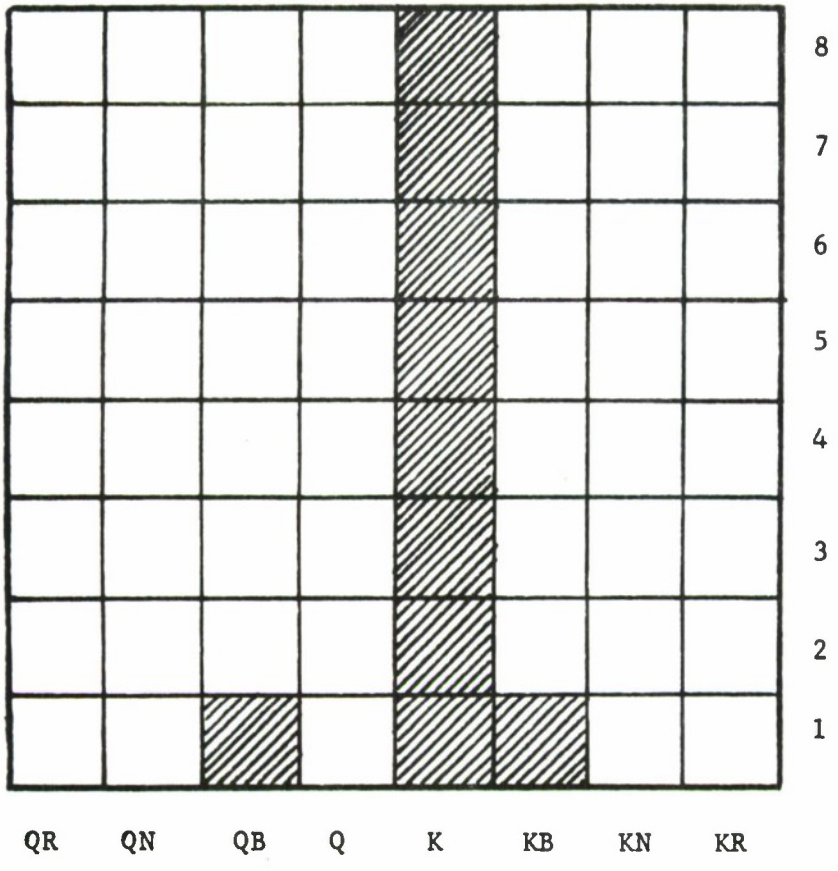


Fig. 10. Combination Logic.

simply supplying it). Each action verb passes an entire data file (or passes data records until a preset limit is sensed) and produces another self-describing data file before the next action verb is called in by the Executive routine.

With this data flow in hand, the COLINGO CONTROL LANGUAGE may be used in programming applications. As illustrated from the data flow, its major difference from normal programming language is the file dependence of the language as opposed to the record dependence of the normal programming language. As shown in Fig. 11, each action verb processes an entire file or sub-file before passing control to the next verb in the message. This file dependent operation is, of course, occasioned by the fact that only one action verb at a time can be contained in core (approximately 12K of the 16K core are overhead programs). The control characters in the individual records are used for tag purposes.

Let us examine this data flow as applied to a previous query.

GET MISSILE-FILE IF TYPE EQ AJAX OR ZEUS CHANGE RANGE TO UPDATED RANGE.

The first action verb to execute is the GET verb, which subsets the data base to locate the MISSILE-FILE. The next action verb to work is IF, which tags all data records containing ZEUS or AJAX, creating a sub-file on tape drive 6 with all the records in the MISSILE-FILE, but with those containing ZEUS or AJAX having tag characters in certain of the 16 control characters appended to each COLINGO tape record.

Finally, the CHANGE verb operates, updating only those records which are tagged, and simply copying the others. Thus, the entire message causes a file to be located, tagged, and processed, in that order.

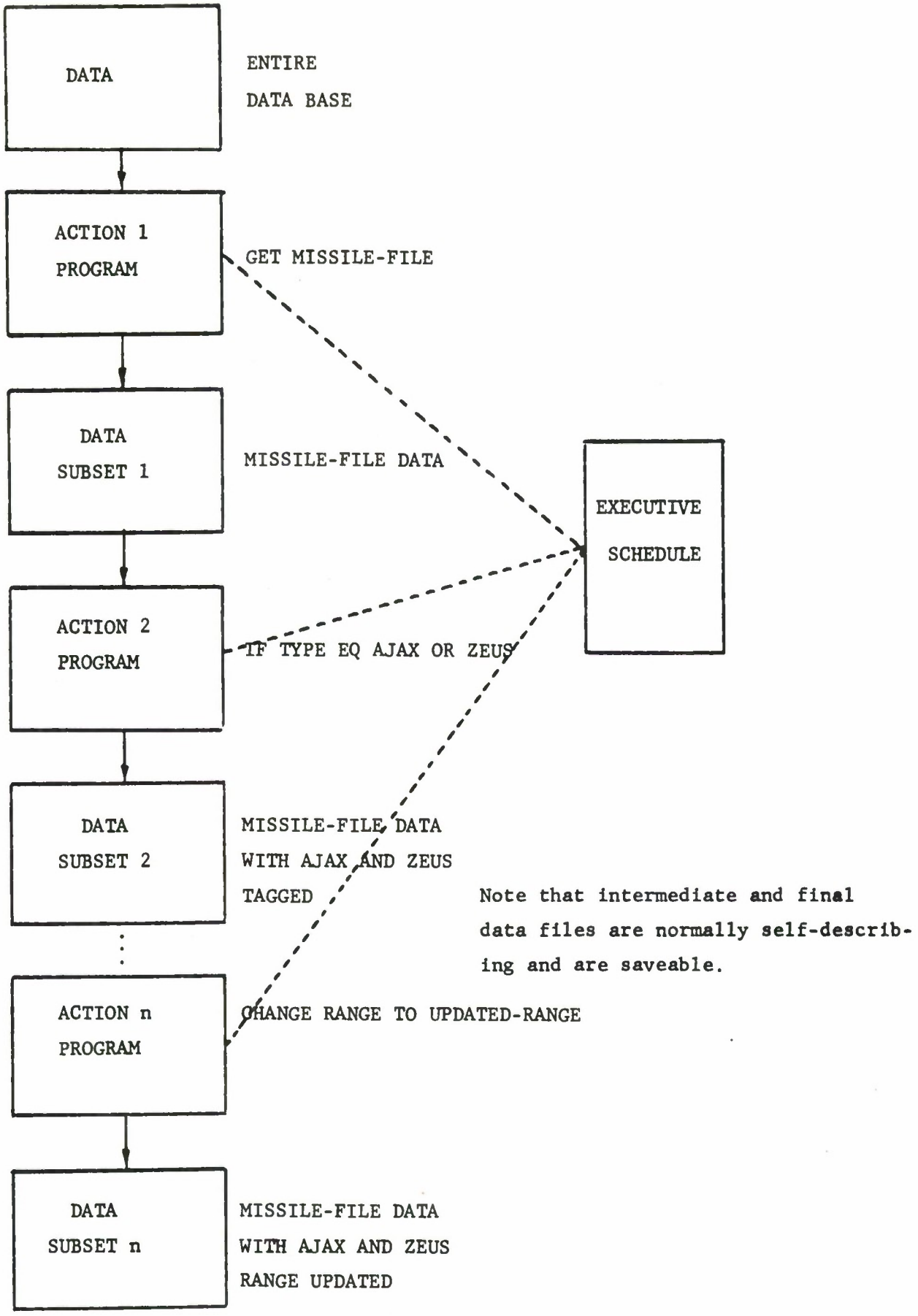


Fig. 11. Data Flow.

The vehicle for producing a COLINGO program is a chained message, with the EXECUTE verb providing the sequencing of messages, and with the IF, IF/NOT verbs providing the conditional branching needed in a programming language.

Chained messages consisting of COLINGO action verbs can be stored on disk and assigned a name or label. The chained message may be called at any time from the console (or from cards or tape) by an "EXECUTE name" statement. This will put COLINGO in the "stored query mode" and will transfer control to the first message at the "name" label. Since all messages in the chain are available on disk at any moment and are labeled, branches from the normal message sequence to other parts of the message chain (or to some other message chain such as a library subroutine) are allowable. Thus, the "instructions" that comprise a COLINGO program are stored on disk, with the Executive module controlling their call-in, and with sequencing under the control of that same Executive. Hence, programs that, if compiled, could take up literally millions of characters of core, can be executed in the COLINGO stored query mode by the above chaining process.

The chained message concept is useful operationally in several areas:

One area is Operator Cues - In an operational process in this application a set of COLINGO messages may be chained together off-line and assigned to disk. Appropriate operator messages may be inserted in the stored messages via the COMMENT verb, which cue the operator according to the process being controlled. Thus, the operator need have no knowledge of CCL in this application.

Another area is Program Library construction - In this application, a number of general purpose processing routines written in CCL may be stored on disk and given a name, and then called by other CCL programs as library routines.

Finally, the COLINGO Executive may be given a looping sequence of CCL routines to execute (such as running certain alarm type or update routines against critical files). These routines will be continually executed unless the operator presses the "REQUEST" key, signalling an on-line operation. Upon completion of the on-line operation, the program may be reset to the "idle loop."

Fig. 12 provides an example of a stored COLINGO message chain. It is called by the input message "EXECUTE EOP.". This will cause the Executive routine to search QUIC file Q for the label name EOP, and to load the message at this label into the core area set aside for messages.

The statement thus loaded is itself an EXECUTE statement, which sets in the Executive routine the schedule of other message labels to be called, and their sequence. Observe that the message at label B1 contains another EXECUTE statement which will be executed before the message at EOPCONT is executed. Once, however, a "multiple" EXECUTE message (i.e., one with several labels following the EXECUTE) is loaded, the Executive routine schedule will be reset with that new message schedule. The single label following an EXECUTE (as in the message at B1) will not reset the Executive routine schedule.

With the data flow and programming aspects of the language in mind, it is pertinent to discuss the actual CCL verbs available.

LABEL	MESSAGE
A0	STATEMENT -- COMMENT 'PLACE THE DICTIONARY CARD DECK IN THE READER' PAUSE COMMENT 'TURN SENSE SWITCH A OFF UNTIL FIRST CARD IS READ' PAUSE.
A1	STATEMENT -- TEST/DIG.
A2	STATEMENT -- GET UTILIZATION-6/2 DICTIONARY
A3	STATEMENT -- GET UTILIZATION-6/2 DICTIONARY DUPLICATE DICTIONARY ON TAPE/BLANK/3.
A4	STATEMENT -- COMMENT 'DICTIONARY IS READY ON TAPE DRIVE 3' COMMENT 'LOAD THE DECK IN THE READER' PAUSE COMMENT ' TURN SENSE SWITCH A OFF UNTIL FIRST CARD IS READ' COMMENT 'TURN SENSE SWITCH C ON UNTIL FILE IS GENERATED'.
A5	STATEMENT -- TEST/GEN.
A6	STATEMENT -- COMMENT 'FILE IS READY ON TAPE DRIVE 2'.
A7	STATEMENT -- COMMENT 'KEY DATA FIELDS WILL NOW BE ANALYZED FOR THEIR CONTENT AND ACCURACY'.
A8	STATEMENT -- GET UTILIZATION-6 ANALYZE PROJECT-NO DEPT JOB-ID PROGRAMMER PRINTER REPORT.
A9	STATEMENT -- PAUSE.
B1	STATEMENT -- COMMENT 'A SERIES OF QUERIES ON CARDS WILL NOW BE READ IN' COMMENT 'THEIR JOB WILL BE TO COMPUTE AN ERROR CHECKING COLUMN' PAUSE (START) EXECUTE B2.
B2	STATEMENT -- SET CARD/IN.
B3	STATEMENT -- GET UTILIZATION-6/5 IF DEPT EQ D78 REPORT/USAGE DEPT.
C1	STATEMENT -- GET UTILIZATION-6 IF TYPE EQ A WRITE/BLANK/3 GET FILE/3 SORT PROGRAMMER.
C2	STATEMENT -- GET FILE/5 TITLE IS '1410/7010 UTILIZATION BY PROGRAMMER' DTG IS 'PAGE 1 OF 5' RIGHT JUSTIFIED CLASSIFICATION IS 'MITRE CONTROLLED' PRINT 1* PROGRAMMER DEPT 3* JOB-ID.
EOP	STATEMENT -- EXECUTE A0 A1 A2 A3 A4 A9 A5 A6 A7 A8 B1 EOPCONT.
EOPCONT	STATEMENT -- EXECUTE A9 B3 A9 C1 C2.

Fig. 12. Stored Message Chain.

COLINGO Action Verbs

The COLINGO action verbs comprise the entire set of legal operations that can be performed on the data files. The action verb set is of course, open-ended, in that special programs may be allocated to the system by name, and these programs then become part of the action verb set.

Fig. 13 illustrates the so-called Basic Program Set (BPS); the set of action verbs that comprises a minimal COLINGO.

COLINGO Input/Output Action Verbs

The Input/Output Action verbs govern the flow of data through the COLINGO system: Although data flow is for the most part automatic, operator control verbs are necessary to specify the file to be processed and the disposition of the generated results.

The GET verb is used to specify the file to be processed, and the dictionary it is to be processed against. Only one file may be opened at a time; however, multi-file processing is possible through the use of a HOLD verb, which shall be explained later.

The TYPEOUT verb is identical to the PRINT verb, except output is to the console typewriter.

The REPORT verb allows non-standard output to be sent to the printer. The non-standard output format must be stored in the computer via an ALLOCATE verb (described later).

The WRITE verb allows data to be output to tape in part or in total to form another system data file.

The DUPLICATE verb allows the duplication of system dictionaries and data.

1.	INPUT/OUTPUT	GET
		PRINT
		TYPE
		PUNCH
		REPORT
		WRITE
		DUPLICATE
		TITLE IS
		CLASSIFICATION IS
		DTG IS
2.	DATA MANIPULATION & COMPUTATION	IF
		IF/NOT
		COMPUTE
		HOLD
		SORT
3.	FILE GENERATION	GENERATE
		DIG
		ANALYZE
4.	FILE UPDATE	UPDATE
		CHANGE
5.	SEQUENCE CONTROL	EXECUTE
6.	MISCELLANEOUS	COMMENT
		PAUSE
7.	MAINTENANCE	ADD
		DELETE
		UPDATE
		SET
		CLEAR
		ALLOCATE
		CREATE

Fig. 13. COLINGO Basic Program Set.

The TITLE IS - CLASSIFICATION IS - DTG IS verbs allow the operator to title hard copy or punch card output with titles, security classification, and dates.

COLINGO Data Manipulation and Computation Action Verbs

The Data Manipulation & Computation Verbs allow complete logical and mathematical control over all fields in the data records which are part of the file specified in the GET statement.

The IF verb allows logical combinations of fields as afforded by the logical operators set forth in Fig. 14. Those records that "qualify" (i.e., are logically true) according to the IF phrase are automatically tagged for further processing or output.

The IF/NOT verb allows for an alternate action verb or alternate message to be executed if no data records qualify according to the IF phrase. This is the conditional branching verb that is critical in a programming language.

The COMPUTE verb allows for mathematical combinations of data fields with each other or with literals according to the mathematical operators, set forth in Fig. 14. It also allows for the creation of new data fields, if desired, to carry the results of computations.

The HOLD verb enables the user to save entire or partial data records from several files which can be processed against the file specified by the GET verb.

The SORT verb enables the user to sort his intermediate or final results on one or more property-name keys.

LOGICAL OPERATORS

AND	
OR	
GR	(Greater Than)
LS	(Less Than)
GQ	(Greater Than or Equal To)
LQ	(Less Than or Equal To)
EQ	(Equal To)
NQ	(Not Equal To)

MATHEMATICAL OPERATORS

$\sqrt{}$	(Root)	SIN	(SINE)
+	(Plus)	COS	(COSINE)
-	(Minus)	ARCOS	(ARC COSINE)
/	(Divide)	ARCSN	(ARC SINE)
*	(Multiply)	GCD	(Great Circle Distance)
**	(Exponentiate)	(	(Left Parenthesis)
		)	(Right Parenthesis)

Fig. I4. COLINGO Logical and Mathematical Operators.

COLINGO File Generation Action Verbs

In the Data Generation area, the user must be able to rapidly insert data into the system (GENERATE verb) and to define it for message processing (DIG verb). In addition, a means must be provided to check the data for errors and discrepancies (ANALYZE verb).

The GENERATE verb provides a flexible program for accepting most fixed-field card files and for accepting certain unit record files on tape. The GENERATE program appends necessary COLINGO control fields to each data record. Thus, external files cannot be processed directly by CCL; they must be first passed through the GENERATE program.

The DIG verb provides the description of the file produced by the GENERATE program. It is possible to describe a file with as many as 36 different dictionaries.

The ANALYZE verb provides the user with a data checking tool to investigate whether data is formatted in accordance with the describing dictionary. It checks such things as data justification and data classification (numeric or alpha) and compares them against the dictionary description. An optional output from the ANALYZE program is an alphabetized listing of data by property-name.

COLINGO Update Action Verbs

It is necessary to provide a means to insert new data without regenerating the file or to effect changes or deletions to existing data. These capabilities are provided by the Update Programs.

The UPDATE verb allows for exception reports or complete change reports based upon one or more objects in a data file. It also provides for deletion of objects or addition of new objects to the file.

The CHANGE verb is especially useful in the area of re-defining existing values in terms of new representations for those values. For example, if regulation specifies a new class of readiness conditions (say F1 for combat readiness instead of C1), this verb would facilitate rapid change of the affected values. As such, this is especially useful in the non - object update application.

COLINGO Sequence Control Action Verbs

A Sequence Verb must be available in any programming language to provide loop control within a program (in the COLINGO case, within a stored message chain).

In the case of CCL, the EXECUTE verb provides this capability. EXECUTE is the action verb that controls the scheduling of messages in the stored message chain. Two levels of scheduling are allowed:

(1) a main level that specifies the normal message sequence (similar to the Instruction Counter in a computer, except that EXECUTE in this application need not specify sequential message labels).

(2) a sub-level that allows deviation from the overall message schedule specified in (1) above.

COLINGO Miscellaneous Action Verbs

A set of Miscellaneous Action Verbs is provided to allow a higher-language mode of operation, i.e., a mode tailored to an operational user performing a specific, pre-planned function.

In this higher-language mode, a message chain is composed using COMMENT verbs, which are the sole method of communication with the

operational user. The actions indicated by the COMMENT verbs cue the operational user through his process, and completely remove him from the general purpose COLINGO verbs. In this same application, the PAUSE verb suspends program execution awaiting an operator action.

COLINGO Maintenance Action Verbs

A set of Maintenance Action Verbs must be provided to the user for bookkeeping the system files and programs, as well as for debugging purposes. System bookkeeping is accomplished through self-describing files called QUIC files, which are normally stored on disk, (although they are also accessible when stored on tape, with degraded response). These files contain system file and program assignments, output masks, and stored message chains. A set of operations is available on these files, as follows:

The ADD verb allows the addition of a dictionary term and/or data value to a QUIC file.

The UPDATE verb is required before an ADD or DELETE verb can be executed, and identifies the QUIC file to be operated upon.

A set of action verbs is provided for control of peripheral devices and for logging and tracing of messages and execution, as follows:

The SET verb enables message input from cards or tape, as well as directing message output to a variety of peripheral devices. It also allows actions such as setting the current date into the system.

The CLEAR verb allows the user to reset any indicator the SET verb has activated. It also allows core to be cleared, excluding the Executive routine.

The ALLOCATE verb allows the assignment of new programs and/or output masks to the COLINGO system. It is the verb that provides for user expansion of the CCL, as determined by his own special needs.

The CREATE verb allows the user to reproduce the COLINGO system for an external user, or for backup protection. It also permits the resulting tape to be reloaded onto disk, perhaps at a different location than that from which it was created.

COLINGO Extension Procedures

Now that the so-called COLINGO Basic Program Set (BPS) has been set forth, it is reasonable to ask how the CCL can be extended in response to actual operational requirements. Most new programs can be allocated to the COLINGO System, and can then be called through the CCL by an assigned name. The Executive module simply extracts the name from the CCL message, loads the named program and its parameters, and transfers control to it. The new program is altered only to the extent that instead of halting, it returns control to the Executive routine upon completion (a Branch instruction to location 15510). Since these non-BPS (i.e., non-Basic Program Set) programs need preserve only the COLINGO Executive routine area (15500-15999), it is relatively convenient to prepare special purpose programs to process COLINGO files. In this case, BPS programs can subset and format data for the non-BPS programs and transfer control to them through the CCL. The only requirements levied on the non-BPS programs are to input and output their own data and to ultimately transfer control back to the Executive routine (location 15510). If the non-BPS programs output COLINGO self-describing data, BPS programs can further process the data.

COLINGO is extremely flexible in accommodating operational program requirements. In many cases, the capability can be provided through a COLINGO BPS Program of chained messages. Some requirements can be met more efficiently with special purpose programs, which can also be allocated and used with COLINGO as described.

COLINGO Output Processing

The ability to output data after processing is of major importance. A standard COLINGO output program is available for normal usage, providing output to printer, console or punch. The standard output produces vertical strings of data values, with automatic horizontal overflow and justification procedures. Although the standard output program (PRINT) provides the most rapid output, its formatting is somewhat restrictive, being limited to outputting vertical data strings of Fig. 15 format, with control over horizontal and vertical spacing on the output strings, (e.g., PRINT HEADING-1 HEADING-2...HEADING-m).

To provide further output flexibility, a user may specify, through a coded report generator sheet (a matrix printer page, which must be keypunched and assigned to COLINGO by name) most other desired output formats. These report forms, when entered and assigned names, are called report masks. A mask may contain all static information, all variable information, or a combination of either. In the "all variable" case, the mask is essentially a sieve, through which may be output data from any file (e.g., REPORT/mask-name HEADING-1, HEADING-2,...HEADING-m). A horizontal data string output format is illustrated in Fig. 16.

Thus, the user can essentially establish general purpose output formats which are either file independent, or which are file dependent (dependent by virtue of having static information entered which is peculiar to the file being processed). Note carefully that once the masks are designed and allocated to the COLINGO System they are available for output usage by means of an on-line verb (REPORT/mask-name).

In addition to the PRINT & REPORT output verbs, the user can also create special purpose (non-BPS) output routines.

HEADING 1	HEADING 2	----------	HEADING m
VALUE 1,1	VALUE 1,2		VALUE 1,m
VALUE 2,1	VALUE 2,2		VALUE 2,m
VALUE 3,1	VALUE 3,2		VALUE 3,m
VALUE 4,1	VALUE 4,2		VALUE 4,m
.	.		.
.	.		.
.	.		.
VALUE n,1	VALUE n,2		VALUE n,m

Fig. 15. Normal Output.

HEADING 1	VALUE 1,1	VALUE 2,1	----	VALUE n,1
HEADING 2	VALUE 1,2	VALUE 2,2	----	VALUE n,2
HEADING 3	VALUE 1,3	VALUE 2,3	----	VALUE n,3
.	.	.		.
.	.	.		.
.	.	.		.
HEADING m	VALUE 1,m	VALUE 2,m	----	VALUE n,m

Fig. 16. Generator Sample Mask.

SUMMARY

In summary, let us reiterate our design objectives and relate them to the solutions afforded by COLINGO.

First and foremost, COLINGO was to be adaptable to a changing set of requirements. Thus, we undertook to design a system which would be evolutionary, which would be responsive in the following areas:

. Data Compatibility

. On-Line Programming

. Simple Extension

. Good Man/Machine Interface

Data Compatibility

In this area, the COLINGO design achieved a large degree of data and program independence by not referring to data fields directly, but instead through a data describing dictionary. This dictionary was created according to COBOL procedures, but achieved the important plus of being modifiable on-line by the operator.

Simple data generation and verification programs were included in COLINGO to generate a large variety of card and tape data files from external sources, and in addition, the CCL itself was given the power to manipulate and change its own data base as to content and format. Thus, not only could the COLINGO System accept a wide variety of formats, it could also process and output that same data in a variety of different formats.

On-Line Programming

In order for a user to quickly obtain results from a generated data file, a combined query and programming language was provided to

manipulate the data base both logically and mathematically. Additionally, a general purpose output program was provided to present results to the user in formats of his choosing.

Simple Extension

With the advance knowledge that not all operational processing requirements and output formats could be conveniently or efficiently produced by the COLINGO Basic Program Set, a simple allocation scheme for introducing special purpose programs into the COLINGO System was provided. This scheme allowed a high degree of communication between the COLINGO Basic Program (BPS) and the special purpose programs.

Man/Machine Interface

Finally, the COLINGO CONTROL LANGUAGE was structured in a way that provided a convenient query language, learned in a matter of days, to complement a more powerful language useful in many programming applications. The combined language was designed to divorce the user as far as possible from the computer equipment configuration and to remove the usual core-size limitations on the operational problem being programmed. This removal was accomplished by the process of chaining operational statements together and allocating them to disk, and having the chain sequentially called in by a small in-core Executive routine.

In retrospect, we have found considerable merit in the general purpose program approach on small computers, and conclude that any inherent inefficiencies introduced by overhead programs would be largely overcome in a larger computer. We feel justified in this conclusion, in that while the overhead program loading of COLINGO amounted to 75% of total core, the actual core consumed by overhead was only 12K characters

on the 1401. Projecting this 12K overhead to a larger core, it is reasonable to assume that any general purpose program inefficiencies would be correspondingly reduced.

REFERENCES

1. IBM 1401 Bulletin: Input/Output Control System, Form J24-1462.
2. IBM COBOL General Information Manual, Form F28-8053-2.

Security Classification

DOCUMENT CONTROL DATA - R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)	2a. REPORT SECURITY CLASSIFICATION
The MITRE Corporation Bedford, Massachusetts	Unclassified
	2b. GROUP

3. REPORT TITLE

The COLINGO System Design Philosophy

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*

N/A

5. AUTHOR(S) *(Last name, first name, initial)*

Spitzer, James F., Robertson, Joseph G. Jr., and Neuse, Durwood H.

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
November 1964	52	2

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
AF 19(628)-2390 b. PROJECT NO. 492 c. d.	ESD-TDR-64-159
	9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)* SR-126

10. AVAILABILITY/LIMITATION NOTICES

Qualified requestors may obtain from DDC.
DDC release to OTS authorized.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	473 & 492L Hq. USAF USSTRICOM Command-Control System L. G. Hanscom Field, Bedford, Mass.

13. ABSTRACT

This report describes the design and operation of COLINGO (Compile On-LINe and GO), a program system embodying a computer control and query language that provides the operator with a grammar and vocabulary approximating English to control program data and equipment in a data processing system. COLINGO provides both logical and mathematical control of data in the system, and allows the operator to specify the output format and content. Additionally, it provides the capability of on-line programming in a higher-ordered language called the COLINGO Control Language (CCL). The system provides internal editing, detection, and error correction features.

A primary design objective was to make COLINGO adaptable to a changing set of requirements and particularly responsive to data compatibility, on-line programming, simple extension, and good man-machine interface. This necessitated maintaining independence of data from programs so that changes in one do not affect the other. The objective was achieved by means of a common retrieval and control program that uses a set of dictionary tables as the translation medium between programs and data. With this concept, changes in programs or data are reflected in modifications to the tables instead of the usual modification to the data or programs, or both.

DD FORM 1 JAN 64 1473

Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Programming Languages Programming (Computers) Digital Computers						

INSTRUCTIONS

1. ORIGINATING ACTIVITY: Enter the name and address of the contractor, subcontractor, grantee, Department of Defense activity or other organization *(corporate author)* issuing the report.

2a. REPORT SECURITY CLASSIFICATION: Enter the overall security classification of the report. Indicate whether "Restricted Data" is included. Marking is to be in accordance with appropriate security regulations.

2b. GROUP: Automatic downgrading is specified in DoD Directive 5200.10 and Armed Forces Industrial Manual. Enter the group number. Also, when applicable, show that optional markings have been used for Group 3 and Group 4 as authorized.

3. REPORT TITLE: Enter the complete report title in all capital letters. Titles in all cases should be unclassified. If a meaningful title cannot be selected without classification, show title classification in all capitals in parenthesis immediately following the title.

4. DESCRIPTIVE NOTES: If appropriate, enter the type of report, e.g., interim, progress, summary, annual, or final. Give the inclusive dates when a specific reporting period is covered.

5. AUTHOR(S): Enter the name(s) of author(s) as shown on or in the report. Enter last name, first name, middle initial. If military, show rank and branch of service. The name of the principal author is an absolute minimum requirement.

6. REPORT DATE: Enter the date of the report as day, month, year; or month, year. If more than one date appears on the report, use date of publication.

7a. TOTAL NUMBER OF PAGES: The total page count should follow normal pagination procedures, i.e., enter the number of pages containing information.

7b. NUMBER OF REFERENCES: Enter the total number of references cited in the report.

8a. CONTRACT OR GRANT NUMBER: If appropriate, enter the applicable number of the contract or grant under which the report was written.

8b, 8c, & 8d. PROJECT NUMBER: Enter the appropriate military department identification, such as project number, subproject number, system numbers, task number, etc.

9a. ORIGINATOR'S REPORT NUMBER(S): Enter the official report number by which the document will be identified and controlled by the originating activity. This number must be unique to this report.

9b. OTHER REPORT NUMBER(S): If the report has been assigned any other report numbers *(either by the originator or by the sponsor)*, also enter this number(s).

10. AVAILABILITY/LIMITATION NOTICES: Enter any limitations on further dissemination of the report, other than those imposed by security classification, using standard statements such as:

(1) "Qualified requesters may obtain copies of this report from DDC."

(2) "Foreign announcement and dissemination of this report by DDC is not authorized."

(3) "U. S. Government agencies may obtain copies of this report directly from DDC. Other qualified DDC users shall request through

_____."

(4) "U. S. military agencies may obtain copies of this report directly from DDC. Other qualified users shall request through

_____."

(5) "All distribution of this report is controlled. Qualified DDC users shall request through

_____."

If the report has been furnished to the Office of Technical Services, Department of Commerce, for sale to the public, indicate this fact and enter the price, if known.

11. SUPPLEMENTARY NOTES: Use for additional explanatory notes.

12. SPONSORING MILITARY ACTIVITY: Enter the name of the departmental project office or laboratory sponsoring *(paying for)* the research and development. Include address.

13. ABSTRACT: Enter an abstract giving a brief and factual summary of the document indicative of the report, even though it may also appear elsewhere in the body of the technical report. If additional space is required, a continuation sheet shall be attached.

It is highly desirable that the abstract of classified reports be unclassified. Each paragraph of the abstract shall end with an indication of the military security classification of the information in the paragraph, represented as *(TS), (S), (C), or (U)*.

There is no limitation on the length of the abstract. However, the suggested length is from 150 to 225 words.

14. KEY WORDS: Key words are technically meaningful terms or short phrases that characterize a report and may be used as index entries for cataloging the report. Key words must be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location, may be used as key words but will be followed by an indication of technical context. The assignment of links, rules, and weights is optional